Universidade Federal do Ceará

Estrutura de Dados Avançada

Arthur Antunes Nogueira da Silva
Afonso Barbosa de Souza Neto
Matrículas: 368334, 369581

Relatório e Análise de Dados

Tabela com todos os resultados dos algoritmos de acordo com a amostragem passada;
Operações / Quantidade de instâncias;

- **Lista Não Ordenada**

Op/QtdInstancias	100	10000	50000	100000	500000	800000
Operações I	2 ms	165 ms	2176 ms	9440 ms	228939 ms	562465 ms
Operações A	1 ms	94 ms	2202 ms	19802 ms	454427 ms	1062119 ms
Operações R	0 ms	90 ms	2193 ms	9705 ms	239597 ms	564550 ms
Operações S	1 ms	117 ms	2733 ms	11331 ms	295019 ms	708895 ms

- **Lista Ordenada**

Op/QtdInstancias	100	10000	50000	100000	500000	800000
Operações I	2 ms	279 ms	4759 ms	20872 ms	512210 ms	1348610 ms
Operações A	2 ms	168 ms	4046 ms	16827 ms	426119 ms	1090159ms
Operações R	1 ms	113 ms	2780 ms	10556 ms	269769 ms	693989ms
Operações S	1 ms	116 ms	2786 ms	11834 ms	298279ms	759095 ms

- **HeapMáximo**

Op/QtdInstancias	100	10000	50000	100000	500000	800000
Operações I	1 ms	152 ms	1349 ms	5343 ms	166423 ms	398672 ms
Operações A	1 ms	62 ms	1329 ms	5271 ms	149408 ms	452730 ms
Operações R	1 ms	52 ms	896 ms	3389 ms	87400 ms	257455 ms
Operações S	1 ms	80 ms	1700 ms	7631 ms	218245 ms	509914 ms

- **Análise de dados**

Para que possamos começar nossa análise, a tabela a seguir vai mostrar a complexidade de cada operação de acordo com os algoritmos testados;

Implementação	Seleção	Inser.	Rem.	Alter.
Lista Não Ordenada	O(n)	O(1)	O(n)	O(n)
Lista Ordenada	O(1)	O(n)	O(1)	O(n)
Heap	O(1)	O(logn)	O(logn)	O(logn)

- **Lista não Ordenada**

Primeiro ponto que devemos observar com todas as informações dos testes disponíveis é que com o número pequeno de instâncias a diferença de cada operação, no algoritmo *Lista Não Ordenada,* é muito pequena e nem sempre a complexidade melhor prevalece.

Op/QtdInstancias	100	10000
Operações I	2 ms	165 ms
Operações A	1 ms	94 ms
Operações R	0 ms	90 ms
Operações S	1 ms	117 ms

Com 100 instâncias, a variância foi de 0 a 2 milissegundos, e na primeira linha, *Operações I,* onde 40% das operações são de inserção foi onde apresentou o maior tempo sendo que é onde deveria ser a menor já que seu grau de complexidade é O(1).

Porém quando o número de instância aumenta consideravelmente, podemos ver que começa aumentar a diferença de tempo;

50000	100000	500000	800000
2176 ms	9440 ms	228939 ms	562465 ms
2202 ms	19802 ms	454427 ms	1062119 ms
2193 ms	9705 ms	239597 ms	564550 ms
2733 ms	11331 ms	295019 ms	708895 ms

Ou seja, quanto maior o número de instâncias, numa implementação de lista não ordenada, onde 40% das operações feitas são de inserção, menor tempo vai levar do que quando a maioria das operações são de outro tipo.

Comparando a Lista Não Ordenada com uma Lista Ordenada e uma implementação HeapMáximo, nas operações de 40% inserção, quando o número de instâncias não é tão grande, o Heap e a Lista Não Ordenada ficam próximos, com instâncias maiores o Heap ganha. Já a Lista Ordenada O(n) sempre fica muito maior;

Lista não ordenada;

Op/QtdInstancias	100	10000	50000	100000	500000	800000
Operações I	2 ms	165 ms	2176 ms	9440 ms	228939 ms	562465 ms

Lista Ordenada;

Op/QtdInstancias	100	10000	50000	100000	500000	800000
Operações I	2 ms	279 ms	4759 ms	20872 ms	512210 ms	1348610 ms

HeapMáximo;

Op/QtdInstancias	100	10000	50000	100000	500000	800000
Operações I	1 ms	152 ms	1349 ms	5343 ms	166423 ms	398672 ms

- **Lista Ordenada**

Quando comparamos os resultados da tabela da implementação, é notório que as operações com 40% remoção e seleção tem um tempo bem menor que os outros em todos números de instâncias;

Op/QtdInstancias	100	10000	50000	100000	500000	800000
Operações I	2 ms	279 ms	4759 ms	20872 ms	512210 ms	1348610 ms
Operações A	2 ms	168 ms	4046 ms	16827 ms	426119 ms	1090159ms
Operações R	1 ms	113 ms	2780 ms	10556 ms	269769 ms	693989ms
Operações S	1 ms	116 ms	2786 ms	11834 ms	298279ms	759095 ms

Como a complexidade dessas operações é O(1), é realmente esperado que o resultado seja esse, em instâncias menores de 100 e 10000, o tempo dobra quase em todos os casos, quando o número de instâncias aumenta muito, a diferença também é quase o dobro em alguns espaços amostrais.

Porém quando a gente compara a Lista Ordenada com as outras implementações, esse número diminui e é superado, até mesmo quando comparamos a Lista não ordenada;

- **Lista Ordenada**

Op/QtdInstancias	100	10000	50000	100000	500000	800000
Operações R	1 ms	113 ms	2780 ms	10556 ms	269769 ms	693989ms
Operações S	1 ms	116 ms	2786 ms	11834 ms	298279ms	759095 ms

- **Lista Não Ordenada**

Op/QtdInstancias	100	10000	50000	100000	500000	800000
Operações R	0 ms	90 ms	2193 ms	9705 ms	239597 ms	564550 ms
Operações S	1 ms	117 ms	2733 ms	11331 ms	295019 ms	708895 ms

- **HeapMáximo**

Op/QtdInstancias	100	10000	50000	100000	500000	800000
Operações R	1 ms	52 ms	896 ms	3389 ms	87400 ms	257455 ms
Operações S	1 ms	80 ms	1700 ms	7631 ms	218245 ms	509914 ms

No Heap é esperado que isso aconteça em grandes instâncias, já que sua complexidade é O(log n) nas operações de remoção mas com 10000 instâncias ele já supera até nas operações de seleção. O curioso realmente é o Lista Não Ordenada conseguir superar ele em todos os números de instâncias passadas.

- **HeapMáximo**

Como esperado o Heap foi melhor que os outros dois quando a instância já estava em 10000, em 100 instâncias quase não apresentou diferença. Como também era esperado na operação de Seleção a partir de 10000 instâncias, o tempo dessa operação é maior que o dos outros 3 tipos de operações já que O(logn) é melhor quando a gente cresce muito o valor o campo amostral de uma população de instâncias. Ou seja, a única análise que podemos tirar dele é que os resultados foram melhores comparado aos outros e a partir de 10000 instâncias a seleção apresenta tempo maior, porém ainda se mostra melhor que as outras implementações.

Op/QtdInstancias	100	10000	50000	100000	500000	800000
Operações I	1 ms	152 ms	1349 ms	5343 ms	166423 ms	398672 ms
Operações A	1 ms	62 ms	1329 ms	5271 ms	149408 ms	452730 ms
Operações R	1 ms	52 ms	896 ms	3389 ms	87400 ms	257455 ms
Operações S	1 ms	80 ms	1700 ms	7631 ms	218245 ms	509914 ms

Para melhor mostrar a diferença e eficiência de cada algoritmo fiz uma tabela que mostra o tempo médio de cada implementação, somando todas as operações e instâncias.

Fica claro de ver com o gráfico a melhor eficiência apresentada pelo Heap dadas as instâncias e as operações.

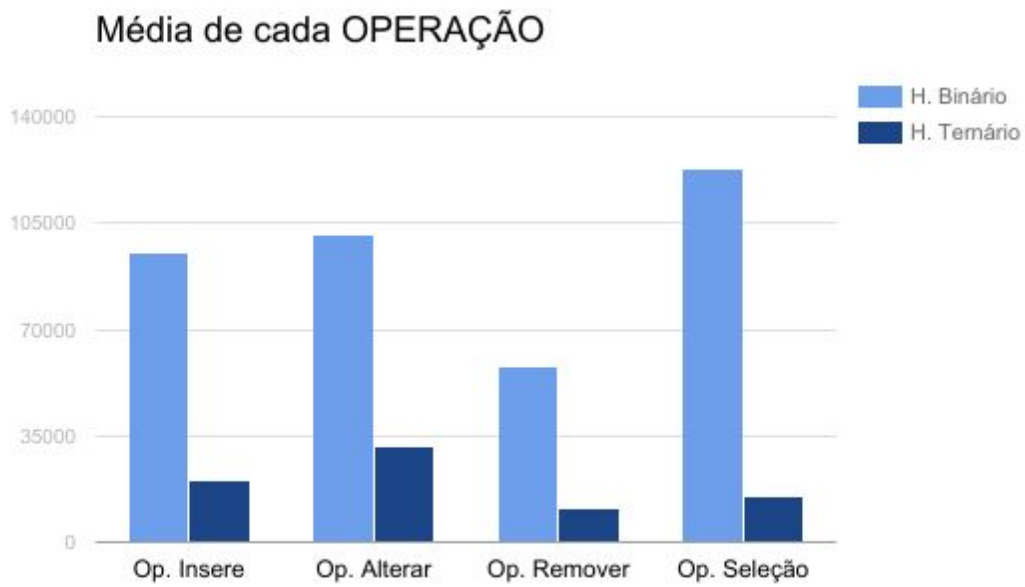
- **Análise Heap Ternário x Heap Binário**

- **Heap Ternário**

Op/QtdInstancias	100	10000	50000	100000	500000	800000
Operações I	1 ms	160 ms	352 ms	1361 ms	35155 ms	86813 ms
Operações A	1 ms	29 ms	563 ms	2017 ms	64389 ms	122178 ms
Operações R	1 ms	15 ms	213 ms	802 ms	19650 ms	45534 ms
Operações S	1 ms	14 ms	322 ms	1168 ms	28195 ms	60691 ms

O Heap Binário, que a gente chamou somente de Heap ou *Heap Máximo* anteriormente, como já diz seu próprio nome, tem apenas 2 “filhos”, que são subjacentes relacionado ao nó pai, mas adjacentes entre eles. Num Heap Ternário, esse número de nós filhos aumenta para 3, ou seja, apesar de não mudar a complexidade do Heap Binário, se opera agora com O(log n), porém na base 3, e os resultados são muito superiores.

Tirando o caso 1 com 100 instâncias, todos os outros testes tem médias muito melhores como vamos poder ver nos gráficos abaixo;



Lembrando que, o gráfico se refere a média de tempo em ms de cada tipo de operação das instâncias passadas pelo professor.

Como podemos ver a discrepância é imensa, na próxima página vamos poder olhar uma a uma e ter uma base da diferença de acordo com o gráfico.

	H. Binário	H. Ternário
Op. Insere	95323,33333	20640,33333
Op. Alterar	101466,8333	31529,5
Op. Remover	58198,83333	11035,83333
Op. Seleção	122928,5	15065,16667

No Insere quase ¼ melhor, no alterar quase ⅓ melhor e no Remover e Seleção, quase ⅛ melhor, isso apenas mudando a base do O(log n) que opera em cima dos algoritmos. Vendo a melhora expressiva que foi mostrada e analisada do Ternário pro Binário, posso afirmar que quanto mai

- **HEAP BINÁRIO E MÉDIAS**

Op/QtdInstancias	100	10000	50000	100000	500000	800000	Média p/ tipo de op	Média GERAL
Operações I	1	152	1349	5343	166423	398672	95323,33333	
Operações A	1	62	1329	5271	149408	452730	101466,8333	
Operações R	1	52	896	3389	87400	257455	58198,83333	
Operações S	1	80	1700	7631	218245	509914	122928,5	
Média p/ n° instâncias	1	86,5	1318,5	5408,5	155369	404692,75		94479,375

- **HEAP TERNÁRIO E MÉDIAS**

Op/QtdInstancias	100	10000	50000	100000	500000	800000	Média p/ tipo de op	Média GERAL
Operações I	1	160	352	1361	35155	86813	20640,33333	
Operações A	1	29	563	2017	64389	122178	31529,5	
Operações R	1	15	213	802	19650	45534	11035,83333	
Operações S	1	14	322	1168	28195	60691	15065,16667	
Média p/ n° instâncias	1	54,5	362,5	1337	36847,25	78804		19567,70833

Podemos ver também que o os testes com o número de instâncias 800000 instâncias, foi onde apresentou a maior discrepância na média comparando com as outras pois no Heap Ternário teve uma média de 78804 e no Binário 404692.75, ou seja, a média do ternário não chega a representar nem 20% da média de 800000 instâncias do Binário.

- **Descrição da máquina geradora dos resultados;**
  Notebook Asus Z450LA-WX007T
  Processador: Intel® Core™ i5-5200U Dual Core 2.20 GHz com Turbo Max até 2.7 GHz
  Cache: 3 MB
  Sistema operacional: Elementary OS
  Memória: RAM 8 GB DDR3L
  Tipo de memória: DDR3L
  Disco rígido: (HD) 1 TB SATA 5400 RPM
  Placa de vídeo: Intel Corporation Broadwell-U Integrated Graphics (rev 09)